NPS-OR-13-002

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

NEW RESULTS ON A STOCHASTIC DUEL GAME WITH EACH FORCE CONSISTING OF HETEROGENEOUS UNITS

by

Kyle Y. Lin

February 2013

Approved for public release; distribution is unlimited

Prepared for: U.S. Army Training and Doctrine Command (TRADOC)
TRADOC Analysis Center TRAC-Monterey
700 Dyer Road, Monterey, CA 93943-0692

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE ***(DD-MM-YYYY)*** 26-02-2013	**2. REPORT TYPE** Technical Report	**3. DATES COVERED** ***(From-To)*** From 19-11-2012 To 30-4-2013

4. TITLE AND SUBTITLE New Results on a Stochastic Duel Game With Each Force Consisting of Heterogeneous Units	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Kyle Y. Lin	**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-OR-13-002
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Training and Doctrine Command (TRADOC) TRADOC Analysis Center TRAC-Monterey 700 Dyer Road, Monterey, CA 93943-0692	**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Two forces engage in a duel, with each force initially consisting of several heterogeneous units. Each unit can be assigned to fire at any opposing unit, but the kill rate depends on the assignment. As the duel proceeds, each force—knowing which units are still alive in real time—decides dynamically how to assign its fire, in order to maximize the probability of wiping out the opposing force before getting wiped out. It has been shown in the literature that an optimal pure strategy exists for this two-person, zero-sum game, but computing the optimal strategy remained cumbersome because of the game's huge payoff matrix. This paper gives an efficient algorithm to compute the optimal strategy without enumerating the entire payoff matrix, and offers some insights into the special case, when one force has only one unit.

15. SUBJECT TERMS
Stochastic duel, two-person zero-sum game, fire allocation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Kyle Lin
Unclassified	Unclassified	Unclassified	UU	25	**19b. TELEPHONE NUMBER** ***(include area code)*** (831) 656-2648

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RDML Jan E. Tighe
Interim President

O. Douglas Moses
Acting Provost

The report entitled *"New Results on a Stochastic Duel Game With Each Force Consisting of Heterogeneous Units"* was prepared for and funded by U.S. Army Training and Doctrine Command (TRADOC), TRADOC Analysis Center TRAC-Monterey, 700 Dyer Road, Monterey, CA 93943-0692.

Further distribution of all or part of this report is authorized.

This report was prepared by:

Kyle Y. Lin
Associate Professor
Department of Operations Research

Reviewed by:

Ronald D. Fricker
Associate Chairman for Research
Department of Operations Research

Robert F. Dell
Chairman
Department of Operations Research

Released by:

Jeffrey D. Paduan
Vice President and Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Two forces engage in a duel, with each force initially consisting of several heterogeneous units. Each unit can be assigned to fire at any opposing unit, but the kill rate depends on the assignment. As the duel proceeds, each force—knowing which units are still alive in real time—decides dynamically how to assign its fire, in order to maximize the probability of wiping out the opposing force before getting wiped out. It has been shown in the literature that an optimal pure strategy exists for this two-person zero-sum game, but computing the optimal strategy remained cumbersome because of the game's huge payoff matrix. This paper gives an efficient algorithm to compute the optimal strategy without enumerating the entire payoff matrix, and offers some insights into the special case, when one force has only one unit.

THIS PAGE INTENTIONALLY LEFT BLANK

1 Introduction

We consider a stochastic duel model with each force consisting of heterogeneous units. Suppose, at the beginning, force A has m units and force B has n units. If A's unit i fires at B's unit j, then the time to kill follows an exponential distribution with rate λ_{ij}. If B's unit j fires at A's unit i, then the time to kill follows an exponential distribution with rate θ_{ji}. If multiple units fire at the same target, then the time to kill follows an exponential distribution, with the rate equal to the sum of individual kill rates. Each force keeps perfect knowledge when a unit gets killed and decides dynamically how to assign its remaining units to fire at the opposing force's remaining units. The goal of each force is to maximize the probability of wiping out the opposing force before getting wiped out.

This stochastic duel model was first studied by Kikuta (1986), and it was shown that a pure optimal strategy exists. It is, however, rather cumbersome to determine the optimal strategy, because one needs to enumerate a huge payoff matrix. Our main contribution in this paper is to establish a necessary and sufficient condition for a pure strategy to be optimal, and use the condition to facilitate an efficient algorithm to compute an optimal strategy. We also provide some insights into the special case, when one force has only one unit.

Two special cases of the model have been reported in the literature. If each force has homogeneous units, such that $\lambda_{ij} = \lambda$ and $\theta_{ji} = \theta$ for all i, j, then any policy that keeps all units busy firing at any opposing unit is optimal. Let $V(m, n)$ denote A's win probability if A has m units and B has n units, for $m, n = 1, 2, \ldots$. A recursive equation can be derived by conditioning on whose unit is killed next, and is given by

$$V(m,n) = \frac{m\lambda}{m\lambda + n\theta} V(m, n-1) + \frac{n\theta}{m\lambda + n\theta} V(m-1, n),$$

with the boundary conditions $V(m, 0) = 1$ for $m \geq 1$, and $V(0, n) = 0$ for $n \geq 1$. Letting $r = \lambda/\theta$, Brown (1963) showed that

$$V(m,n) = r^n \sum_{k=1}^{m} \frac{(-1)^{m-k}\, k^{m+n}\, \Gamma(rk+1)}{(m-k)!\, k!\, \Gamma(n+rk+1)}.$$

When the units are heterogeneous, it makes a difference how each force allocates his fire. In addition, the fire allocation may change as both forces lose their units during the duel.

Another special case, when $m = 1$, was previously studied by Friedman (1977) and Kikuta (1983), where A needs to determine which fire order, among the $n!$ possible fire orders, is optimal. In many sequencing problems, when one decides in which order to process a number of jobs, it is possible to compute an index for each job based on its own attributes, and to obtain the optimal sequence by sorting those indices (Ross, 1983; Gittins et al., 2011). Unfortunately, in this problem the preference between two targets depends on the other targets still alive, which makes the problem difficult. Friedman (1977) gave a necessary condition for the optimal order, while Kikuta (1983) strengthened the necessary condition and gave a sufficient condition for optimality. In general, however, to find the optimal fire order one needs to compare all $n!$ fire orders by brute force.

The study of duel models dates back to the 1910s, when Lanchester (1916) proposed differential equations that govern the strength of each force through time, which gave rise to what later became known as Lanchester models. A stream of works extended the Lanchester models—which are deterministic in nature—to stochastic duel models by introducing randomness to shot outcomes, time between taking shots, etc.; see, for instance, Brown (1963); Williams and Ancker (1963); Barfoot (1974); Kress (1992); and Kress and Talmor (1999). These stochastic duel models, however, assume homogeneous units, so there is no decision making. The focus of earlier works was to obtain expressions for win probability in various duel scenarios. Readers interested in comprehensive surveys on combat models are referred to Ancker (2006); Washburn and Kress (2009); and Kress (2012).

The rest of this paper proceeds as follows. Section 2 presents the main results, where we give a necessary and sufficient condition for a pure strategy to be optimal, and then use the condition to facilitate an efficient algorithm to compute the optimal strategy. Section 3 discusses the special case when $m = 1$, and gives a condition under which the preference between two targets can be readily determined, *regardless* of the other targets still alive.

2 Main Results

At the beginning, force A (or player A) has a set of units $S_A = \{1, 2, \ldots, m\}$ and force B (or player B) has a set of units $S_B = \{1, 2, \ldots, n\}$. As the duel proceeds, each player keeps real-time knowledge about when a unit gets killed. In other words, each player has full information about the history of the game and, at any time point, decides how to allocate his fire on the opponent's remaining units. Because we assume exponential kill rates, knowing which units are still alive on both sides, the future of the game becomes independent from its past.

At any time point, the state of the duel can be delineated by (S'_A, S'_B), with $S'_A \subseteq S_A$ being the set of A's remaining units, and $S'_B \subseteq S_B$ the set of B's remaining units. The game belongs to the class of Markov games, because once in a state, the previous actions and results become irrelevant to the future of the game. It is also an exhaustive game according to the definition in Washburn (2003), because each state will be visited once at most. For a given state (S'_A, S'_B), the two players can be viewed as playing a single-stage game, which ends as soon as any unit on either side is killed. In other words, by letting $V(S'_A, S'_B)$ denote A's win probability in state (S'_A, S'_B), then the payoff to A is $V(S'_A \setminus \{i\}, S'_B)$ if A's unit i is killed next, and is $V(S'_A, S'_B \setminus \{j\})$ if B's unit j is killed next. In addition, because the player loses the game if he loses all his units, we have that $V(S'_A, \emptyset) = 1$, if $S'_A \neq \emptyset$, and $V(\emptyset, S'_B) = 0$, if $S'_B \neq \emptyset$. Consequently, if we can solve this single-stage game, then we can compute the optimal strategy recursively on $|S'_A| + |S'_B|$, beginning from $1, 2, \ldots$, and so on.

The rest of this section focuses on the single-stage game. Section 2.1 recounts how to construct a single-stage game in matrix form, as was done in Kikuta (1986). Section 2.2 gives a necessary and sufficient condition for a saddle point in this matrix, and Section 2.3 gives an efficient algorithm to find a saddle point without enumerating the entire payoff matrix.

2.1 Single-Stage Game in Matrix Form

Consider the beginning of the game when the state is (S_A, S_B). For notational convenience, write $a_i \equiv V(S_A \setminus \{i\}, S_B)$ for all $i \in S_A$, and $b_j \equiv V(S_A, S_B \setminus \{j\})$ for all $j \in S_B$. That is, a_i is A's win probability if he loses unit i in state (S_A, S_B), and b_j is A's win probability if he kills B's unit j in state (S_A, S_B).

A pure strategy in state (S_A, S_B) is a fire allocation. For $i \in S_A, j \in S_B$, let

$$x_{ij} = \begin{cases} 1, & \text{if } A\text{'s unit } i \text{ fires at } B\text{'s unit } j, \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ji} = \begin{cases} 1, & \text{if } B\text{'s unit } j \text{ fires at } A\text{'s unit } i, \\ 0, & \text{otherwise.} \end{cases}$$

The set of A's pure strategies is

$$\Pi_A = \left\{ \boldsymbol{x} = [x_{ij}] : x_{ij} \in \{0,1\}, i \in S_A, j \in S_B; \text{ and } \sum_{j \in S_B} x_{ij} = 1, \text{for all } i \in S_A \right\}. \tag{1}$$

Because each of A's m units can fire at any of B's n units, the number of A's pure strategy is $|\Pi_A| = n^m$. Similarly, the set of B's pure strategies is

$$\Pi_B = \left\{ \boldsymbol{y} = [y_{ji}] : y_{ji} \in \{0,1\}, i \in S_A, j \in S_B; \text{ and } \sum_{i \in S_A} y_{ji} = 1, \text{for all } j \in S_B \right\}, \tag{2}$$

with $|\Pi_B| = m^n$.

Given A's pure strategy $\boldsymbol{x}$, let

$$\Lambda_j(\boldsymbol{x}) = \sum_{i \in S_A} x_{ij} \lambda_{ij} \tag{3}$$

denote the rate at which B's unit j gets killed. In other words, the amount of time it takes for A to kill B's unit j follows an exponential distribution with rate $\Lambda_j(\boldsymbol{x})$, if A uses pure strategy $\boldsymbol{x}$. Similarly, if B uses pure strategy $\boldsymbol{y}$, let

$$\Theta_i(\boldsymbol{y}) = \sum_{j \in S_B} y_{ji} \theta_{ji}$$

denote the rate at which A's unit i gets killed.

If A chooses a pure strategy $\boldsymbol{x} \in \Pi_A$, and B chooses a pure strategy $\boldsymbol{y} \in \Pi_B$, then by conditioning on which unit gets killed next, the probability that A will eventually win the duel is given by

$$f(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}) b_j + \sum_{i \in S_A} \Theta_i(\boldsymbol{y}) a_i}{\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}) + \sum_{i \in S_A} \Theta_i(\boldsymbol{y})}, \tag{4}$$

which is also the payoff to A for the pure strategy pair $(\boldsymbol{x}, \boldsymbol{y})$. The payoff to B is $1 - f(\boldsymbol{x}, \boldsymbol{y})$, or equivalently, $-f(\boldsymbol{x}, \boldsymbol{y})$.

In this two-person zero-sum game in standard matrix form, A has n^m pure strategies and B has m^n pure strategies. Kikuta (1986) showed that this matrix game has a saddle point. To determine the saddle point, however, one needed to enumerate the entire payoff matrix of size n^m by m^n.

Remark 1 The two-person zero-sum game discussed in this section can be regarded as a special case of a race-to-reward game as follows. Two players, A and B, each have resources to allocate among tasks. A has a set of resources, S_A, to allocate among a set of tasks, T_A, with allocation of resource i to task k leading to a task completion rate λ_{ik}, for $i \in S_A$ and $k \in T_A$. Similarly, B has a set of resources, S_B, to allocate among a set of tasks, T_B, with allocation of resource j to task l leading to a task completion rate θ_{jl}, for $j \in S_B$ and $l \in T_B$. Each task has an associated reward to A, namely a_k for $k \in T_A$, and b_l for $l \in T_B$, with $a_k > b_l$ for all $k \in T_A$ and $l \in T_B$ to avoid triviality. The payoff to A is the reward of the task that is completed first. The game is zero-sum, with A trying to maximize his expected payoff and B trying to minimize it. If $T_A = S_B$ and $T_B = S_A$, then this race-to-reward game reduces to the single-stage duel game described in this section. Although we present our analysis in the context of a single-stage duel game, all the results can be straightforwardly extended to the race-to-reward game.

2.2 Necessary and Sufficient Condition for Saddle Points

Theorem 1 gives an alternative proof that the matrix game in Section 2.1 has a saddle point. The proof also shows how to determine the optimal strategy if one knows the value of the game, and facilitates a necessary and sufficient condition for a saddle point, which we present in Theorem 2.

Theorem 1 Consider the two-person zero-sum game defined by pure strategy sets Π_A in (1), Π_B in (2), and player A's payoff function $f(\boldsymbol{x}, \boldsymbol{y})$ in (4). This game has at least one saddle point. In particular, letting v^* denote the value of the game, $\boldsymbol{x}' \in \Pi_A$ a pure strategy that maximizes

$$\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}) \cdot (b_j - v^*), \tag{5}$$

and $\boldsymbol{y}' \in \Pi_B$ a pure strategy that minimizes

$$\sum_{i \in S_A} \Theta_i(\boldsymbol{y}) \cdot (a_i - v^*), \tag{6}$$

then $f(\boldsymbol{x}', \boldsymbol{y}') = v^*$, and $(\boldsymbol{x}', \boldsymbol{y}')$ is a saddle point.

Proof. We prove the theorem by contradiction. First, suppose that $\boldsymbol{x}'$ maximizes (5) and $\boldsymbol{y}'$ minimizes (6), but $f(\boldsymbol{x}', \boldsymbol{y}') > v^*$, or equivalently,

$$0 < \sum_{j \in S_B} \Lambda_j(\boldsymbol{x}')(b_j - v^*) + \sum_{i \in S_A} \Theta_i(\boldsymbol{y}')(a_i - v^*). \tag{7}$$

Because $\boldsymbol{y}'$ minimizes (6), it follows that

$$\sum_{i\in S_A}\Theta_i(\boldsymbol{y}')(a_i-v^*)\le\sum_{i\in S_A}\Theta_i(\boldsymbol{y})(a_i-v^*),\qquad\forall\boldsymbol{y}\in\Pi_B.\tag{8}$$

Adding $\sum_{j\in S_B}\Lambda_j(\boldsymbol{x}')(b_j-v^*)$ to both sides of the preceding, together with (7), we can conclude that

$$0<\sum_{j\in S_B}\Lambda_j(\boldsymbol{x}')(b_j-v^*)+\sum_{i\in S_A}\Theta_i(\boldsymbol{y})(a_i-v^*),\qquad\forall\boldsymbol{y}\in\Pi_B,$$

or equivalently,

$$f(\boldsymbol{x}',\boldsymbol{y})>v^*,\qquad\forall\boldsymbol{y}\in\Pi_B.$$

In other words, using the pure strategy $\boldsymbol{x}'$, player A can guarantee a payoff strictly greater than v^*, showing that the value of the game is strictly greater than v^*, which is a contradiction that v^* is the value of the game.

Second, by supposing that $f(\boldsymbol{x}',\boldsymbol{y}')<v^*$, we can draw a similar contradiction. Therefore, we have shown that $f(\boldsymbol{x}',\boldsymbol{y}')=v^*$.

To prove that $(\boldsymbol{x}',\boldsymbol{y}')$ is a saddle point, we need to show that $f(\boldsymbol{x}',\boldsymbol{y})\ge v^*$ for all $\boldsymbol{y}\in\Pi_B$, and $f(\boldsymbol{x},\boldsymbol{y}')\le v^*$ for all $\boldsymbol{x}\in\Pi_A$. To do so, note that

$$\begin{aligned}0&=\sum_{j\in S_B}\Lambda_j(\boldsymbol{x}')(b_j-v^*)+\sum_{i\in S_A}\Theta_i(\boldsymbol{y}')(a_i-v^*),\\&\le\sum_{j\in S_B}\Lambda_j(\boldsymbol{x}')(b_j-v^*)+\sum_{i\in S_A}\Theta_i(\boldsymbol{y})(a_i-v^*),\qquad\forall\boldsymbol{y}\in\Pi_B,\end{aligned}$$

where the equality follows from $f(\boldsymbol{x}',\boldsymbol{y}')=v^*$, and the inequality from adding $\sum_{j\in S_B}\Lambda_j(\boldsymbol{x}')(b_j-v^*)$ to (8). Hence, $f(\boldsymbol{x}',\boldsymbol{y})\ge v^*$ for all $\boldsymbol{y}\in\Pi_B$. A similar argument shows that $f(\boldsymbol{x},\boldsymbol{y}')\le v^*$ for all $\boldsymbol{x}\in\Pi_A$. Consequently, $(\boldsymbol{x}',\boldsymbol{y}')$ is a saddle point. □

If we know the value of the game v^*, then, according to Theorem 1, the optimal strategy for A is $\boldsymbol{x}$, which maximizes

$$\sum_{j\in S_B}\Lambda_j(\boldsymbol{x})\cdot(b_j-v^*)=\sum_{j\in S_B}\sum_{i\in S_A}\lambda_{ij}x_{ij}\cdot(b_j-v^*)=\sum_{i\in S_A}\left(\sum_{j\in S_B}x_{ij}\cdot\lambda_{ij}(b_j-v^*)\right),\tag{9}$$

where $\Lambda_j(\boldsymbol{x})$ is defined in (3). Once v^* is known, each of A's units can determine which opposing unit to fire at separately. For A's unit i, he should simply compare $\lambda_{ij}(b_j-v^*)$ for all $j\in S_B$ and find the largest value. In other words, it is optimal for A's unit i to fire at B's unit j^*, where

$$j^*=\arg\max_{j\in S_B}\lambda_{ij}(b_j-v^*).\tag{10}$$

In case of a tie, break it arbitrarily, in which case there will be multiple optimal pure strategies and multiple saddle points. The optimal policy is to set $x_{ij^*}=1$, and $x_{ij}=0$ for $j\ne j^*$.

It follows immediately from (10) that, if $\lambda_{i_1j}=\lambda_{i_2j}$ for all j, then there exists an optimal strategy, with which A's units i_1 and i_2 fire at the same target. This result strengthens Corollary 2 in Kikuta (1986), which requires $\theta_{ji_1}=\theta_{ji_2}$ for all j.

Theorem 2 A pair of pure strategies $(\boldsymbol{x}', \boldsymbol{y}')$ is a saddle point, and a real number v' is the value of the game, if and only if all three conditions hold:

C1. $\boldsymbol{x}'$ maximizes $\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}) \cdot (b_j - v')$,

C2. $\boldsymbol{y}'$ minimizes $\sum_{i \in S_A} \Theta_i(\boldsymbol{y}) \cdot (a_i - v')$,

C3. $f(\boldsymbol{x}', \boldsymbol{y}') = v'$.

Proof. From Theorem 1, the game has at least one saddle point. Denote by $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ a saddle point, and v^* the value of the game. It follows immediately that $f(x^*, y^*) = v^*$.

To prove that C1–C3 are sufficient conditions, we need to show $v' = v^*$. To prove $v' = v^*$ by contradiction, first suppose that $v' < v^*$ to get a string of inequalities involving $\boldsymbol{x}'$ and $\boldsymbol{x}^*$ as

$$\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}')(b_j - v') \geq \sum_{j \in S_B} \Lambda_j(\boldsymbol{x}^*)(b_j - v') > \sum_{j \in S_B} \Lambda_j(\boldsymbol{x}^*)(b_j - v^*),$$

where the first inequality follows from C1, while the second inequality follows because of the assumption $v' < v^*$. Similarly, we get another string of inequality involving $\boldsymbol{y}'$ and $\boldsymbol{y}^*$ as

$$\sum_{i \in S_A} \Theta_i(\boldsymbol{y}')(a_i - v') > \sum_{i \in S_A} \Theta_i(\boldsymbol{y}')(a_i - v^*) \geq \sum_{i \in S_A} \Theta_i(\boldsymbol{y}^*)(a_i - v^*),$$

where the first inequality follows because of the assumption $v' < v^*$, while the second inequality follows from Theorem 1. Adding these two equations together, we arrive at

$$\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}')(b_j - v') + \sum_{i \in S_A} \Theta_i(\boldsymbol{y}')(a_i - v') > \sum_{j \in S_B} \Lambda_j(\boldsymbol{x}^*)(b_j - v^*) + \sum_{i \in S_A} \Theta_i(\boldsymbol{y}^*)(a_i - v^*). \quad (11)$$

The left-hand side of the preceding is 0 according to C3, while the right-hand side is also 0 since $f(x^*, y^*) = v^*$. Hence, we arrive at a contradiction.

If we suppose $v' > v^*$ instead, then we can use a similar argument to draw a contradiction. Consequently, we have shown that $v' = v^*$. Finally, using Theorem 1, together with $v' = v^*$, C1, and C2, it follows that $(\boldsymbol{x}', \boldsymbol{y}')$ is a saddle point. Therefore, we have proved that C1–C3 are sufficient conditions.

We next prove that C1–C3 are necessary conditions. To prove C3, we write

$$f(\boldsymbol{x}', \boldsymbol{y}') = v^* = v',$$

where the first equality follows because $(\boldsymbol{x}', \boldsymbol{y}')$ is a saddle point, and the second follows because v' is the value of the game.

To prove C1 and C2, note that because $(\boldsymbol{x}', \boldsymbol{y}')$ is a saddle point, $f(\boldsymbol{x}', \boldsymbol{y}')$ must be the smallest in its row and largest in its column. The former implies that

$$f(\boldsymbol{x}', \boldsymbol{y}') \leq f(\boldsymbol{x}', \boldsymbol{y}), \qquad \forall \boldsymbol{y} \in \Pi_B,$$

with equality when $\boldsymbol{y} = \boldsymbol{y}'$. Use C3 to replace the left-hand side with v', and use (4) to spell out the right-hand side. After some algebra, the preceding equation becomes

$$\sum_{j \in S_B} \Lambda_j(\boldsymbol{x}') \cdot (b_j - v') + \sum_{i \in S_A} \Theta_i(\boldsymbol{y}) \cdot (a_i - v') \geq 0, \quad \forall \boldsymbol{y} \in \Pi_B,$$

with equality when $\boldsymbol{y} = \boldsymbol{y}'$. In other words, $\boldsymbol{y}'$ minimizes $\sum_{i \in S_A} \Theta_i(\boldsymbol{y}) \cdot (a_i - v')$, which proves C2. Beginning with

$$f(\boldsymbol{x}', \boldsymbol{y}') \geq f(\boldsymbol{x}, \boldsymbol{y}'), \qquad \forall \boldsymbol{x} \in \Pi_A,$$

with equality when $\boldsymbol{x} = \boldsymbol{x}'$, we can use a similar argument to prove C1. Consequently, we have proved that C1–C3 are necessary conditions. □

2.3 Computing Saddle Points

This section presents an iterative algorithm to compute saddle points without enumerating the entire payoff matrix of size $n^m \times m^n$. The algorithm goes as follows.

1. Pick v arbitrarily in $[0, 1]$.

2. For $v \in [0, 1]$, define

$$\hat{\boldsymbol{x}}(v) \equiv \arg\max_{\boldsymbol{x}} \sum_{j \in S_B} \Lambda_j(\boldsymbol{x}) \cdot (b_j - v), \tag{12}$$

$$\hat{\boldsymbol{y}}(v) \equiv \arg\min_{\boldsymbol{y}} \sum_{i \in S_A} \Theta_i(\boldsymbol{y}) \cdot (a_i - v). \tag{13}$$

   In case of a tie, break it arbitrarily. Next, compute

$$T(v) \equiv f(\hat{\boldsymbol{x}}(v), \hat{\boldsymbol{y}}(v)) = \frac{\sum_{j \in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v)) b_j + \sum_{i \in S_A} \Theta_i(\hat{\boldsymbol{y}}(v)) a_i}{\sum_{j \in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v)) + \sum_{i \in S_A} \Theta_i(\hat{\boldsymbol{y}}(v))}. \tag{14}$$

3. If $T(v) = v$, then v is the value of the game and $(\hat{\boldsymbol{x}}(v), \hat{\boldsymbol{y}}(v))$ is a saddle point. If $T(v) \neq v$, then update $v \leftarrow T(v)$, and go to step 2.

It is worth noting that computing $\hat{\boldsymbol{x}}(v)$ and $\hat{\boldsymbol{y}}(v)$ in (12) and (13) does not require linear programming, and can be done quickly, as is the case in (10). When the algorithm stops, we have a triplet $(\hat{\boldsymbol{x}}(v), \hat{\boldsymbol{y}}(v), v)$ that satisfies the three conditions in Theorem 2; therefore, the optimal solution. It follows immediately from Theorem 2 that the value of the game v^* is a fixed point of the function $T(\cdot)$, namely $T(v^*) = v^*$. We next present two lemmas, before proving that the algorithm will stop after a finite number of iterations. Although Lemma 1 can be viewed as a special case of Lemma 2, we put them separately for ease of explanation.

Lemma 1 If $v < v^*$, then $T(v) > v$; if $v > v^*$, then $T(v) < v$.

Proof. From Theorem 1 there exists a saddle point; let $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ denote one. If $v < v^*$, then using the same argument that gives rise to (11), we have that

$$\sum_{j \in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v))(b_j - v) + \sum_{i \in S_A} \Theta_i(\hat{\boldsymbol{y}}(v))(a_i - v) > \sum_{j \in S_B} \Lambda_j(\boldsymbol{x}^*)(b_j - v^*) + \sum_{i \in S_A} \Theta_i(\boldsymbol{y}^*)(a_i - v^*).$$

The right-hand side of the preceding is equal to 0, because $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = v^*$. Therefore,

$$\frac{\sum_{j \in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v)) b_j + \sum_{i \in S_A} \Theta_i(\hat{\boldsymbol{y}}(v)) a_i}{\sum_{j \in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v)) + \sum_{i \in S_A} \Theta_i(\hat{\boldsymbol{y}}(v))} > v,$$

or equivalently, $T(v) > v$. If $v > v^*$, then, using a similar argument, we can show that $T(v) < v$, which completes the proof. $\square$

Let $T^{(0)}(v) \equiv v$, and for $k = 1, 2, \ldots$, let $T^{(k)}(v) \equiv T \circ T^{(k-1)}(v)$. The next lemma generalizes Lemma 1.

Lemma 2 If $v < v^*$, then $T^{(k)}(v) > v$, for $k = 1, 2, \ldots$; if $v > v^*$, then $T^{(k)}(v) < v$, for $k = 1, 2, \ldots$.

Proof. Consider the case $v < v^*$, and for notational simplicity write $v_0 \equiv v < v^*$, and $v_k \equiv T^{(k)}(v)$, for $k = 1, 2, \ldots$. We need to show that $v_k > v_0$ for $k = 1, 2, \ldots$.

Because $v_0 < v^*$, it follows from Lemma 1 that $v_1 > v_0$. If $v_1 < v^*$, then it follows from Lemma 1 again that $v_2 > v_1 > v_0$. In other words, the sequence $v_0, v_1, \ldots$ increases strictly until, at some point, it either reaches v^* or exceeds v^*. In the former case, all following numbers in the sequence are v^* because $T(v^*) = v^*$, so it is true that $v_k > v_0$ for $k = 1, 2, \ldots$.

Suppose now that the sequence $v_0, v_1, \ldots$ exceeds v^* at some point. Let

$$s = \min\{k : v_k > v^*\}.$$

In other words, $v_0 < v_1 < \cdots < v_{s-1} < v^* < v_s$, as depicted in Figure 1. Because $v_s > v^*$, it follows again from Lemma 1 that $v_{s+1} < v_s$. If $v_k \geq v^*$ for $k = s, s+1, \ldots$, then the statement that $v_k > v_0$ for all $k = 1, 2, \ldots$ is also true.

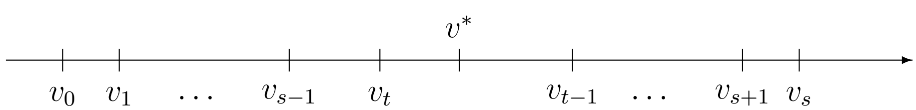

Figure 1: This diagram depicts the sequence $v_0, v_1, \ldots, v_s, \ldots, v_t, \ldots$, where $v_k \equiv T^{(k)}(v_0)$, for $k = 0, 1, \ldots$. Each new number in the sequence either gives a better lower bound or a better upper bound for v^*, and the sequence converges to v^* after a finite number of iterations.

To complete the proof, suppose now that the sequence $v_s, v_{s+1}, \ldots$ drops below v^* at some point, and let

$$t = \min\{k : k > s, v_k < v^*\}.$$

In other words, $v_t < v^* < v_{t-1} < \cdots < v_s$; as depicted in Figure 1. We next show that $v_t > v_{s-1}$. To do so, write a string of inequality

$$\begin{aligned}\sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{t-1}))(b_j - v_{s-1}) &> \sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{t-1}))(b_j - v_{t-1})\\ &\geq \sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{s-1}))(b_j - v_{t-1})\\ &> \sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{s-1}))(b_j - v_s),\end{aligned}$$

where the first inequality follows because $v_{s-1} < v^* < v_{t-1}$; the second inequality follows from the definition of $\hat{\boldsymbol{x}}(v_{t-1})$; the third inequality follows because $v_{t-1} < v_s$. Similarly, write another string of inequality

$$\begin{aligned}\sum_{i\in S_A} \Theta_i(\hat{\boldsymbol{y}}(v_{t-1}))(a_i - v_{s-1}) &\geq \sum_{i\in S_A} \Theta_i(\hat{\boldsymbol{y}}(v_{s-1}))(a_i - v_{s-1})\\ &> \sum_{i\in S_A} \Theta_i(\hat{\boldsymbol{y}}(v_{s-1}))(a_i - v_s),\end{aligned}$$

where the first inequality follows from the definition of $\hat{\boldsymbol{y}}(v_{s-1})$, and the second inequality follows because $v_{s-1} < v^* < v_s$. Adding these two inequalities gives

$$\begin{aligned}&\sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{t-1}))(b_j - v_{s-1}) + \sum_{i\in S_A} \Theta_i(\hat{\boldsymbol{y}}(v_{t-1}))(a_i - v_{s-1})\\ &> \sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{s-1}))(b_j - v_s) + \sum_{i\in S_A} \Theta_i(\hat{\boldsymbol{y}}(v_{s-1}))(a_i - v_s).\end{aligned}$$

The right-hand side of the preceding is equal to 0, because $v_s = T(v_{s-1})$. Hence, we arrive at

$$\sum_{j\in S_B} \Lambda_j(\hat{\boldsymbol{x}}(v_{t-1}))(b_j - v_{s-1}) + \sum_{i\in S_A} \Theta_i(\hat{\boldsymbol{y}}(v_{t-1}))(a_i - v_{s-1}) > 0,$$

or equivalently,

$$\frac{\sum_j \Lambda_j(\hat{\boldsymbol{x}}(v_{t-1}))b_j + \sum_i \Theta_i(\hat{\boldsymbol{y}}(v_{t-1}))a_i}{\sum_j \Lambda_j(\hat{\boldsymbol{x}}(v_{t-1})) + \sum_i \Theta_i(\hat{\boldsymbol{y}}(v_{t-1}))} > v_{s-1}.$$

Because the left-hand side of the preceding is just $T(v_{t-1}) = v_t$, we conclude that $v_t > v_{s-1} > v_0$. By repeating this argument, we can see that $v_k > v_0$, for all $k = 1, 2, \ldots$. The case of $v_0 > v^*$ can be proved in a similar fashion. □

Theorem 3 The algorithm will stop after a finite number of iterations.

Proof. Recall that, in the game matrix, A has n^m pure strategies (rows) and B has m^n pure strategies (columns). There are, at most, $n^m \times m^n$ distinct payoff values in the game matrix,

each of which corresponds to $f(\boldsymbol{x}, \boldsymbol{y})$ for some pure strategy pair $(\boldsymbol{x}, \boldsymbol{y})$. In addition, at least one of the payoff values is v^*, since the game has a saddle point.

Again for notational simplicity, write $v_0 \equiv v$, and $v_k \equiv T^{(k)}(v)$, for $k = 1, 2, \ldots$. To prove the theorem, suppose instead that the algorithm does not stop, or equivalently, $v_k \neq v^*$ for all $k = 0, 1, 2, \ldots$. Because $v_k \neq v^*$, it follows from Lemma 2 that its value will not be repeated in the subsequence $v_{k+1}, v_{k+2}, \ldots$. In other words, all numbers in the sequence $v_0, v_1, v_2, \ldots$ are distinct. Other than v_0, however, each number in the sequence $v_1, v_2, \ldots$ corresponds to a payoff value in the game matrix. We then arrive at a contradiction because there are only a finite number of distinct payoff values in the game matrix, which completes the proof. □

The proof in Theorem 3 shows that the algorithm will stop after at most $n^m \times m^n$ iterations. This worst case would happen if all the payoff values in the game matrix are distinct, and if the sequence $T^{(k)}(v)$, $k = 1, 2, \ldots$ visits all these distinct values. In practice, the actual number of iterations required to compute v^* is often far smaller than $n^m \times m^n$, because the sequence gets closer to v^* after each iteration. In particular, as seen from the proof in Lemma 2, each new value generated in the sequence is either v^*, or the best lower bound to date if it is less than v^*, or the best upper bound to date if it is larger than v^*.

One way to speed up the computation is to pick the initial value close to v^* to reduce the number of iterations. To this end, note that $\max_{i \in S_A} a_i \leq v^* \leq \min_{j \in S_B} b_j$, because A will increase his win probability if he kills any of B's units, and decrease his win probability if any of his units are killed. Hence, an initial pick between $\max_{i \in S_A} a_i$ and $\min_{j \in S_B} b_j$, such as

$$v = \frac{1}{2}\left(\max_{i \in S_A} a_i + \min_{j \in S_B} b_j\right),$$

should work well.

This algorithm can be used to recursively compute $V(S_A, S_B)$, the value of the game in state (S_A, S_B). Specifically, we need to compute $V(S'_A, S'_B)$ for all $S'_A \subseteq S_A$ and $S'_B \subseteq S_B$, by iterating on $|S'_A| + |S'_B|$, the total number of units still alive. The case when $|S'_A| + |S'_B| = 1$ is trivial. If we have computed $V(S'_A, S'_B)$ for all states when $|S'_A| + |S'_B| = k$, then those values become the a_i and b_j used to compute v^* for states (S'_A, S'_B) with $|S'_A| + |S'_B| = k+1$, which, in turn, becomes the a_i and b_j for next iteration when $|S'_A| + |S'_B| = k + 2$.

Example 1 Consider an example with three unit types: rock (R), paper (P), and scissors (S). Assume that

$$\begin{aligned}
\lambda_{\mathrm{R,P}} &= \lambda_{\mathrm{P,S}} = \lambda_{\mathrm{S,R}} = 0.5, \\
\lambda_{\mathrm{R,R}} &= \lambda_{\mathrm{P,P}} = \lambda_{\mathrm{S,S}} = 1, \\
\lambda_{\mathrm{R,S}} &= \lambda_{\mathrm{S,P}} = \lambda_{\mathrm{P,R}} = 2,
\end{aligned}$$

and $\theta_{ij} = \lambda_{ij}$ for $i, j = \mathrm{R, P, S}$. Table 1 gives the probability that A wins the duel in various states. For instance, if A has 2 rocks and B has 1 rock and 1 paper, then A will win the duel with probability $V(\mathrm{RR}, \mathrm{RP}) = 0.262$.

There is one interesting observation. Whereas, in 1-on-1 and 2-on-2 duels, the win probability depends highly on the unit types on each side, in a 3-on-3 duel having RPS

Table 1: Probability that Player A wins the duel in different states as discussed in Example 1, when there are three unit types: Rock, Paper, and Scissors.

	Player B								
Player A	PP	RP	RS	PS	RPS	RPP	RSS	PPS	PSS
R	0.022	0.071	0.320	0.133	0.040	0.007	0.178	0.013	0.076
RR	0.111	0.262	0.696	0.375	0.186	0.049	0.533	0.079	0.279
RP	0.304	0.500	0.623	0.377	0.228	0.186	0.358	0.121	0.189
RRR	0.270	0.494	0.906	0.609	0.402	0.152	0.813	0.211	0.514
RRP	0.467	0.673	0.879	0.642	0.463	0.325	0.714	0.286	0.453
RRS	0.721	0.618	0.814	0.812	0.474	0.353	0.675	0.547	0.647
RPS	0.814	0.772	0.772	0.772	0.500	0.526	0.537	0.537	0.526

would guarantee a win probability at least 0.5, regardless of the opponent's three units. It is better to have a balanced force, which makes it difficult for the opponent to exploit the weakness. □

3 One Against Many

Consider the special case when $m = 1$. The optimal strategy for B is clearly for all his remaining units to fire at A's only unit, while A needs to decide in which of the $n!$ possible orders his only unit should fire at B's units. Because A has only one unit, in this section we write $\lambda_j = \lambda_{1j}$, and $\theta_j = \theta_{j1}$ for notational convenience. We also use target j and B's unit j interchangeably.

The problem has been previously studied by Friedman (1977) and Kikuta (1983), where they identified necessary conditions and sufficient conditions for an optimal fire order. In particular, Friedman (1977) showed that if the fire order $1, 2, \ldots, n$ is optimal, then

$$\theta_k \left(\lambda_k + \theta_k + \sum_{i=k+2}^{n} \theta_i \right) \geq \theta_{k+1} \left(\lambda_{k+1} + \theta_{k+1} + \sum_{i=k+2}^{n} \theta_i \right), \quad \text{for } k = 1, 2, \ldots, n-1, \tag{15}$$

because otherwise, swapping k and $k+1$ results in a better fire order. Equation (15), however, is not a sufficient condition for optimality, as seen by a counterexample given in Kikuta (1983). There is no simple way to rank the targets in a complete list, because whether target k or target $k+1$ should be fired at first depends on the other targets present, as seen by the term $\sum_{i=k+2}^{n} \theta_i$ in (15).

Intuitively, A prefers to fire at a target that is easier to kill so as to eliminate a target sooner. He should also prefer to kill a target that poses a bigger threat. That is, if $\lambda_1 > \lambda_2$ and $\theta_1 > \theta_2$, then it is intuitive that A should kill target 1 before trying to kill target 2, regardless of the other targets still alive. It turns out this conjecture is true. The next theorem presents

a slightly weaker condition than the preceding one, under which it is possible to rank the preference between two targets, *regardless* of the other targets still alive.

Theorem 4 If either

1. $\theta_1 > \theta_2$ and $\lambda_1 + \theta_1 \geq \lambda_2 + \theta_2$, or
2. $\theta_1 = \theta_2$ and $\lambda_1 > \lambda_2$,

then target 1 stands higher than target 2 in the optimal fire order, regardless of the other targets.

Proof. Consider fire order 1:

$$\ldots, 1, i_1, \ldots, i_k, 2, j_1, \ldots, j_l.$$

Let $\alpha = \sum_{s=1}^{k} \theta_{i_s}$ and $\beta = \sum_{s=1}^{l} \theta_{j_s}$ for notational convenience. The probability that A wins with fire order 1 is

$$\begin{aligned}
&P\{\text{wipe out all targets in front of target 1 before getting killed}\}\\
&\times \frac{\lambda_1}{\lambda_1+\theta_1+\alpha+\theta_2+\beta}\left(\prod_{s=1}^{k}\frac{\lambda_{i_s}}{\lambda_{i_s}+\sum_{r=s}^{k}\theta_{i_r}+\theta_2+\beta}\right)\frac{\lambda_2}{\lambda_2+\theta_2+\beta}\\
&\times P\{\text{wipe out targets } j_1,\ldots,j_l \text{ before getting killed}\}.
\end{aligned} \tag{16}$$

Swap targets 1 and 2 in fire order 1 to get fire order 2:

$$\ldots, 2, i_1, \ldots, i_k, 1, j_1, \ldots, j_l.$$

The probability that A wins with fire order 2 is

$$\begin{aligned}
&P\{\text{wipe out all targets in front of target 2 before getting killed}\}\\
&\times \frac{\lambda_2}{\lambda_2+\theta_1+\alpha+\theta_2+\beta}\left(\prod_{s=1}^{k}\frac{\lambda_{i_s}}{\lambda_{i_s}+\sum_{r=s}^{k}\theta_{i_r}+\theta_1+\beta}\right)\frac{\lambda_1}{\lambda_1+\theta_1+\beta}\\
&\times P\{\text{wipe out targets } j_1,\ldots,j_l \text{ before getting killed}\}.
\end{aligned} \tag{17}$$

The first term and the last term in equations (16) and (17) are identical. In addition, the product term in the middle in (16) is at least as large as the product term in the middle in (17), because of the assumption $\theta_1 \geq \theta_2$ in the theorem. Hence, fire order 1 is strictly better than fire order 2 if

$$\frac{\lambda_1}{\lambda_1+\theta_1+\alpha+\theta_2+\beta}\frac{\lambda_2}{\lambda_2+\theta_2+\beta} > \frac{\lambda_2}{\lambda_2+\theta_1+\alpha+\theta_2+\beta}\frac{\lambda_1}{\lambda_1+\theta_1+\beta}.$$

If either condition stated in the theorem is met, then one can verify that

$$\begin{aligned}
&(\lambda_1+\theta_1+\alpha+\theta_2+\beta)(\lambda_2+\theta_2+\beta)-(\lambda_2+\theta_1+\alpha+\theta_2+\beta)(\lambda_1+\theta_1+\beta)\\
&=\theta_2(\lambda_2+\theta_2)+(\lambda_2+\theta_2)\alpha+\theta_2\beta-\theta_1(\lambda_1+\theta_1)-(\lambda_1+\theta_1)\alpha-\theta_1\beta<0,
\end{aligned}$$

which completes the proof. $\square$

A natural question to ask is whether Theorem 4 can be extended to the case when A has $m \geq 2$ units. In other words, if each of the m units can individually rank all of B's units according to the condition in Theorem 4, then does each unit's optimal fire order collectively give rise to the group optimal policy? It turns out that is not the case, as seen in the counterexample below.

Example 2 Consider an example with $m = 2$ and $n = 2$, with their kill-rate matrices given as follows:

$$[\lambda_{ij}] = \begin{pmatrix} 0.9 & 1 \\ 1 & 0.9 \end{pmatrix}, \qquad [\theta_{ji}] = \begin{pmatrix} 1.1 & 1 \\ 1 & 1.1 \end{pmatrix}.$$

In state $(\{1\}, \{1,2\})$, from the standpoint of A's unit 1, the rates

$$\lambda_{11} = 0.9, \quad \lambda_{12} = 1, \quad \theta_{11} = 1.1, \quad \theta_{21} = 1$$

meet the condition in Theorem 4. Therefore, in state $(\{1\}, \{1,2\})$ it is optimal for A's unit 1 to fire at B's unit 1. For the same reason, in state $(\{2\}, \{1,2\})$ it is optimal for A's unit 2 to fire at B's unit 2.

If A still has both units and B also has both units, however, then A's pure strategy

$$\boldsymbol{x} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

is *not* optimal, as another pure strategy

$$\boldsymbol{x}' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

can do strictly better, because

$$\Lambda_1(\boldsymbol{x}') = 1 > 0.9 = \Lambda_1(\boldsymbol{x}),$$
$$\Lambda_2(\boldsymbol{x}') = 1 > 0.9 = \Lambda_2(\boldsymbol{x}).$$

As a matter of fact, $\boldsymbol{x}'$ is optimal for A in state $(\{1,2\}, \{1,2\})$. Hence, even if each unit can individually rank all opposing units according to the condition in Theorem 4, collectively, these individual rankings do not necessarily give rise to the group optimal policy. $\square$

To conclude this section, we give a condition weaker than the one in Theorem 4, under which the fire order $1, 2, \ldots, n$ is optimal. Theorem 2 in Kikuta (1983) also gives a sufficient condition for the fire order $1, 2, \ldots, n$ to be optimal. It is straightforward to show that Kikuta's condition implies the one in Corollary 1; however, the condition in Corollary 1 is much easier to verify.

Corollary 1 If

$$\theta_1 \geq \theta_2 \geq \cdots \geq \theta_n,$$

and

$$\theta_1(\lambda_1 + \theta_1) \geq \theta_2(\lambda_2 + \theta_2) \geq \cdots \geq \theta_n(\lambda_n + \theta_n),$$

then the fire order $1, 2, \ldots, n$ is optimal.

Proof. Consider an arbitrary fire order $\ldots, i, j, \ldots$ other than $1, 2, \ldots, n$, where $i > j$. For any constant $D \geq 0$, we can verify that

$$\theta_i(\lambda_i + \theta_i + D) \leq \theta_j(\lambda_j + \theta_j + D).$$

Hence, according to (15), swapping i and j results in another fire order that is at least as good.

Starting with an arbitrary fire order, we can repeatedly look for adjacent targets that are out of order and swap them—analogous to bubble sort—so that in each step we get a new fire order that is at least as good. When no such swapping is possible, we arrive at the fire order $1, 2, \ldots, n$, which is at least as good as the initial fire order. Because this argument works for any initial fire order, the fire order $1, 2, \ldots, n$ is optimal. $\square$

Acknowledgment

The authors would like to thank Alan Washburn, Moshe Kress, and Michael Atkinson for their suggestions that improve the presentation of this paper.

References

Ancker, C. J. (2006). A proposed foundation for a theory of combat. *Naval Research Logistics Quarterly*, 42(3):311–343.

Barfoot, C. B. (1974). Markov duels. *Operations Research*, 22(2):318–330.

Brown, R. H. (1963). Theory of combat: The probability of winning. *Operations Research*, 11(3):418–425.

Friedman, Y. (1977). Optimal strategy for the one-against-many battle. *Operations Research*, 25(5):884–888.

Gittins, J., Glazebrook, K. D., and Weber, R. (2011). *Multi-Armed Bandit Allocation Indices*. Wiley, United Kingdom, 2nd edition.

Kikuta, K. (1983). A note on the one against many battle. *Operations Research*, 31(5):952–956.

Kikuta, K. (1986). The matrix game derived from the many-against-many battle. *Naval Research Logistics Quarterly*, 33(4):603–612.

Kress, M. (1992). A many-on-many stochastic duel model for a mountain battle. *Naval Research Logistics*, 39:437–446.

Kress, M. (2012). Modeling armed conflicts. *Science*, 336(6083):865–869.

Kress, M. and Talmor, I. (1999). A new look at the 3:1 rule of combat through markov stochastic lanchester models. *The Journal of the Operational Research Society*, 50(7):733–744.

Lanchester, F. W. (1916). *Aircraft in Warfare: The Dawn of the Fourth Arm*. Constable, London.

Ross, S. M. (1983). *Introduction to Stochastic Dynamic Programming*. Academic Press, New York.

Washburn, A. R. (2003). *Two-Person Zero-Sum Games*. INFORMS, Rockville, MD, 3rd edition.

Washburn, A. R. and Kress, M. (2009). *Combat Modeling*. Springer.

Williams, T. and Ancker, C. J. (1963). Stochastic duels. *Operations Research*, 11(5):803–817.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, California

4. Richard Mastowski (Technical Editor)..1
   Graduate School of Operational and Information Sciences (GSOIS)
   Naval Postgraduate School
   Monterey, California

5. Jonathan K. Alt 1
   TRAC-Monterey
   Monterey, California

6. Christopher E. Marks 1
   TRAC-Monterey
   Monterey, California

7. Christian J. Darken...1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California

8. Arnold H. Buss 1
   Modeling, Virtual Environments, and Simulation Institute
   Naval Postgraduate School
   Monterey, California

9. Kyle Y. Lin 1
   Operations Research Department
   Naval Postgraduate School
   Monterey, California